L Number	Hits	Search Text	DB	Time stamp
5	1	ldap and (xml-schema) and (maintain$4 adj session)	US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/31 00:55
7	0	ldap and (xml adj schema) and (maintain$4 adj session)	USPAT	2003/07/31 00:56
8	1	(ldap and (xml adj schema) and session)	USPAT	2003/07/31 00:57
9	0	ldap and xml-rpc and session	USPAT	2003/07/31 00:57
10	5	ldap and xml-rpc and session	US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/31 00:59
11	44	ldap and xml and rpc and session	US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/31 01:02
12	22	ldap and xml and rpc and session	USPAT	2003/07/31 01:02

L Number	Hits	Search Text	DB	Time stamp
5	1	ldap and (xml-schema) and (maintain$4 adj session)	US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/31 00:55
7	0	ldap and (xml adj schema) and (maintain$4 adj session)	USPAT	2003/07/31 00:56
8	1	(ldap and (xml adj schema) and session)	USPAT	2003/07/31 00:57
9	0	ldap and xml-rpc and session	USPAT	2003/07/31 00:57
10	5	ldap and xml-rpc and session	US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/31 00:59
11	44	ldap and xml and rpc and session	US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/31 01:02
12	22	ldap and xml and rpc and session	USPAT	2003/07/31 01:18
13	35	ldap and xml and schema	USPAT	2003/07/31 01:19

L Number	Hits	Search Text	DB	Time stamp
3	16514	registry and service and ldap interface and session	USPAT	2003/07/31 22:27
4	269	(xml and registry and service and ldap interface and ((save maintain$4 keep$4) adj session))	USPAT	2003/07/31 21:57
5	194	((xml and registry and service and ldap interface and ((save maintain$4 keep$4) adj session))) and @ad<=19990610	USPAT	2003/07/31 21:55
6	20	(schema and xml and (registry database directory) and service and ldap and (interface api rpc) and ((save maintain$4 keep$4) adj session))	USPAT	2003/07/31 21:57

Citation

Symposium on Applied Computing >archive
Proceedings of the 1999 ACM symposium on Applied computing >toc
1999 , San Antonio, Texas, United States

Implementing catalog clearinghouses with XML and XSL

Author
Andrew V. Royappa

Sponsors
SIGADA : ACM Special Interest Group on Ada Programming Language
SIGCUE : ACM Special Interest Group on Computer Uses In Education
SIGAPP : ACM Special Interest Group on Applied Computing
SIGBIO : ACM Special Interest Group on Biomedical Computing

Publisher
ACM Press New York, NY, USA

Pages: 616 - 621 Series-Proceeding-Article
Year of Publication: 1999
ISBN:1-58113-086-4

http://doi.acm.org/10.1145/298151.298495 (Use this link to Bookmark this page)

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Serge Abiteboul , Nicole Bidoit, Non first normal form relations: An algebra allowing data restructuring, Journal of Computer and System Sciences, v.33 n.3, p.361-393, Dec. 1986

2 Alfred V. Aho , Ravi Sethi , Jeffrey D. Ullman, Compilers: principles, techniques, and tools, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1986

3 Berg, D.L., Gonnet, G. H. and Tompa, F. W., The New Oxford English Dictionary Project at the University of Waterloo, Technical Report OED-88-01. University of Waterloo Centre for the New Oxford English Dictionary (1988).

4 Blake G.E., Consens M.P., Kilpel~iinen, P., Larson, P.- A., Snider, T., and Tompa, F.W., Text/Relational Database Management Systems: Harmonizing SQL and SGML, Proceedings of Applications of Databases (ADB-94), Vadstena, 267-280 (1994).

5 Bray, T., Paoili, J., and Sperberg-McQueen, C. M., (ed) Extensible Markup Language (XML) 1.0, World Wide Web Consortium Recommendation 10-February-1998 (1988).

6 Burnard, L., Introducing SARA: an SGML-Aware Retrieval Application for the British National Corpus, Second Conference on Teaching and Language Corpora, Lancaster University (1996).

7 Paolo Cincarini , Alfredo Rizzi , Fabio Vitali, An extensible rendering engine for XML and HTML, Proceedings of the seventh international conference on World Wide Web 7, p.225-237, April 1998, Brisbane, Australia

8 Clark, J. and Deach, S. (ed), Extensible Stylesheet Language (XSL), Version 1.0, World Wide Web Consortium Working Draft 18-August-1998 (1998).

9 Clark, J. XT (Java transformation engine for XSL). http://jclark.com/xml/xt.html (1998).

10 Codd, E.F., Further Normalization of the Data Base Relational Model. in Data Base Systems, Courant Computer Science Symposia Series 6. Englewood Cliffs, NJ, Prentice-Hall (1972).

11 Cover, R., Extensible Markup Language. A very large repository of XML and XSL information, at the Summer blstitute of Linguistics. http://www.sil.org/sgmllxml.html (1997).

12 Cyberchefs Home Page, http://www.cyberchefs.com/ (1998).

13 Cybermeals Home Page, http://www.cybermeals.com/ (1998).

14 Deutsch, A., Fernandez, M., Florescu, D., Levy, A., and Suciu, D., XML-QL: A Query language for XML, Submission to the World Wide Web Consortium 19- August- 1998 (1998).

15 Goldfarb, C. F., A Generalized Approach to Document Markup, ACM SIGPI~4N Notices 16(6), 68-73 (1981).

16 Gonnet, G.H., gaeza-Yates, R. A., Snider, T.: Lexicographical Indices for Text: Inverted files vs. PAT trees, Technical Report OED-91-01. University of Waterloo Centre for the New Oxford English Dictionary (1991).

17 Hafi~, S., Le Maitre, J., Murisasco, E. and V6ronis, J., The MULTEXT SgmlQL Query Language Reference, Multilingual Text Tools and Corpora Project, Centre National de la Recherche Scientifique, fittp://www, lpl.univaix.fr/projects/SgmlQLl (1997).

18 ISO 8879:1986, Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). International Organization for Standardization. Ref. No. ISO 8879:1986 (E). Geneva/New York (1986).

19 ISO/IEC DIS 10179.2:1994. Information Technology- Text and Office Systems - Document Style Semantics and Specification Language (DSSSL). International Organization for Standardization/ International Electrotechnical Commission. Geneva (1994).

20 Kuikka, E. and Nikunen, E., Systems for Structured Documents (large list of software), http ://w w w. cs. uku. fi/--ku i kka/S yste ms97/syste ms97. html (1998).

21 Lie, H. and Bos, B., Cascading Style Sheets, level 1, World Wide Web Consortium Recommendation

17-December-1996 (1996).

22 R. W. Matzen , G. E. Hedrick, A new tool for SGML with applications for the World Wide Web, Proceedings of the 1998 ACM symposium on Applied Computing, p.752-759, February 27-March 01, 1998, Atlanta, Georgia, United States

23 Darrell R. Raymond, Flexible Text Display with Lector, Computer, v.25 n.8, p.49-60, August 1992

24 The Microsoft XSL Processor, Technology Preview Release, Microsoft Corporation, http://www, microsoft.com/xml/xsl/msxsl.asp (1998).

25 XML Styler, ArborText Inc., http://www.arbortext.com/xmlstyler/index.htm (1998).

↑ CITINGS

↑ INDEX TERMS

Primary Classification:
H. Information Systems
↳ **H.3** INFORMATION STORAGE AND RETRIEVAL

General Terms:
Algorithms, Design, Performance

Keywords:
SGML, XML, XSL, e-commerce

↑ Peer to Peer - Readers of this Article have also read:

- MBONE: the multicast backbone
  Communications of the ACM 37, 8
  Hans Eriksson

- Efficient and effective placement for very large circuits
  Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design
  Wern-Jieh Sun , Carl Sechen

- The CAVE: audio visual experience automatic virtual environment
  Communications of the ACM 35, 6
  Carolina Cruz-Neira , Daniel J. Sandin , Thomas A. DeFanti , Robert V. Kenyon , John C. Hart

- Clustering and linear placement
  Proceedings of the ninth design automation workshop on Design automation
  Donald M. Schuler , Ernst G. Ulrich

- Editorial pointers
  Communications of the ACM 44, 9
  Diane Crawford

IMPLEMENTING CATALOG CLEARINGHOUSES WITH XML AND XSL

Andrew V. Royappa
Department of Computer Science
Millsaps College
Jackson, MS 39210
royapav@millsaps.edu

Keywords: XML, XSL, SGML, e-commerce

ABSTRACT

A catalog clearinghouse is defined as an electronic commerce entity that provides a common web-based storefront to a group of merchants. This paper considers the implementation of a specific kind of clearinghouse: one that hosts catalogs for a large number of similar merchants (e.g., thousands of restaurant menus). A standard architecture is described, followed by a novel architecture that uses the emerging Extensible Markup Language (XML) and Extensible Stylesheet Language (XSL) standards for data storage, search and graphical presentation.

For large clearinghouses, the new architecture realizes significant benefits in terms of presentation flexibility, powerful search capabilities and increased performance. XML and XSL are used at the client side for flexible presentations with low maintenance costs, while XML and traditional RDBMS techniques are combined at the server for searching and business logic. Detailed examples of XML document type definitions and XML structured data are given, along with details on the recursive transformation of XML into HTML, using rule-based XSL style sheets.

1. INTRODUCTION

As e-commerce on the WWW intensifies, a number of meta-businesses have emerged. Known by various names such as "virtual mall", such businesses provide a single access point for groups of merchants. Customers access the "catalog clearinghouse" web site to browse catalogs of merchant companies that deliver goods and services. The clearinghouse handles ordering and payment, and transmits orders to merchants for fulfillment. This enables customers to search for goods, compare prices, etc., at a central location instead of visiting multiple dislocated web sites.

SAC '99, San Antonio, Texas
1998 ACM 1-58113-086-4/99/0001 $5.00

The clearinghouse model is also attractive to small businesses for which the benefits of maintaining an individual web site may never justify the overhead.

The model requires little technical expertise on the part of the merchant, and little modification to standard business procedure. For instance, orders can be forwarded from the clearinghouse via fax or telephone. Restaurants fit this category, and it is therefore natural that clearinghouses have appeared which specialize solely in restaurants offering take-out or home delivery services. Among these, *Cyberchefs* [12] and *Cybermeals* [13] include thousands of restaurants. In this paper, a restaurant clearinghouse is used as an illustrative example.

A look at existing clearinghouses, large and small, reveals two broad implementation strategies:

1. The clearinghouse maintains a set of customized web pages for each merchant.

2. The clearinghouse maintains a database containing catalog information for each merchant. When a merchant's page is accessed, its web pages are generated dynamically.

For large clearinghouses that store several thousand catalogs (e.g., *Cybermeals* targets 25,000 restaurants by the end of 1998), only the second approach is feasible. Maintaining and updating thousands of customized web forms (e.g. to make price and product changes) would be prohibitive, even if custom web pages were initially created for each business. However, the second approach has certain drawbacks:

1. All catalogs have an identical "look and feel" because catalog web pages are programmatically generated. This makes it hard for each merchant to establish an individual sales identity.

2. Catalog search capabilities are limited and inefficient. Consider the case of a restaurant clearinghouse that allows each restaurant to list the chief ingredients of its dishes, along with nutrition information such as calories, fat grams, diabetic information, etc. A simple keyword search may not accurately target "a Chinese dish with chicken and snow peas, without peanuts or MSG, with at most 25 grams of fat, within 45 minutes delivery time of Harvard Square in Boston."

If such a query *is* allowed, other problems arise. In any normalized relational database schema [10] constructed to store

such information, this query will require a costly relational join of several tables. A large clearinghouse may expect thousands of such queries per minute at peak periods. Finally, while relational databases can easily search for expressions containing comparison and Boolean operators, they are cumbersome for searches dealing with hierarchical relationships between fields such as "inside/outside", "above/below", and transitive compositions of such relationships.

This paper presents an architecture, or implementation model, that uses *Extensible Markup Language* (XML) [5] and *Extensible Stylesheet Language* (XSL) [8] to overcome the above problems. These are web-based structured document standards that are being developed based on *Standardized Generalized Markup Language* (SGML) [15,18].

The architecture presented allows a clearinghouse to have a unique presentation format for each merchant's catalog, without the overhead of maintaining individual customized web pages. By moving formatting tasks to the client (browser), the scheme provides load balancing that reduces web server workload. Furthermore, expressive and efficient catalog searches are made possible. Finally, the fault tolerance that is realized by using a relational database to drive the business logic (including transaction processing for ordering, payment and delivery) is largely left undisturbed.

The following section contains a brief introduction to XML and its companion, XSL. After that, the paper describes the catalog clearinghouse architecture in detail. Examples of XML document type definitions and structure are given, along with examples of the recursive transformation of XML into ordinary HTML using XSL style sheets. The paper concludes with an overview of software tools for implementation and avenues for future work.

2. MARKUP AND STYLESHEET LANGUAGES

2.1 XML for Structured Document Representation

Standard Generalized Markup Language (SGML) is a document representation standard developed by the International Organization for Standardization, published as ISO 8879 [18]. It is a meta-language for specifying the syntax of domain-specific markup languages. HyperText Markup Language (HTML) is one application of SGML.

SGML contains features that are difficult to learn and implement, although they are not commonly used in practice. Therefore, the World Wide Web Consortium (W3C) has constructed a version of SGML suitable for web use, namely XML [5]. Differences between XML and SGML include the following:

- XML does not allow exceptions (*inclusions* and *exclusions*) in element content models.
- XML does not have the "&" operator that denotes elements which are required but can appear in any order.
- XML tags for elements with non-empty content models must come in pairs, e.g. <a>.... Tags with empty content models have the special form <a/> (a trailing "/" after the tag name).
- XML does not include abbreviation techniques (tag minimization, tag omission).

These changes simplify XML at the cost of expressive power. For instance, exceptions are a powerful expressive feature of SGML, but are difficult for both human readers and automatic parsers to handle because they introduce ambiguity ([22] describes techniques for modeling and understanding exceptions and presents a case for their inclusion in XML).

The structure of XML documents can be described using *Document Type Definitions* (DTD), which are specified in Extended Backus-Naur Form [2]. A DTD is similar to a context free grammar, so standard parsing techniques from compiler theory apply. XML documents can conform to a DTD. They are information rich in the sense that all semantic text is appropriately marked up. By contrast, semantic information in HTML documents has to be extracted from unstructured paragraphs, using natural language processing or heuristic techniques based on keyword positioning.

XML documents generally don't contain presentation information. This is stored elsewhere using a system of structure-based presentation rules (explained below). The clear identification of semantic information via markup elements makes it possible to have expressive user interfaces for searching, e.g [3], and allows for powerful indexing schemes for quickly searching very large textual databases, e.g. [3,6,16].

Both Microsoft and Netscape have announced that version 5 of their respective browsers will contain XML processors, which are software components that parse XML documents and make the resulting tree structures available for manipulation in code.

2.2 XSL for Transformation and Rendering of XML

The W3C has specified an *Extensible Stylesheet Language* (XSL) [8] for presentation and display of XML documents. XSL can rearrange document structure, making it strictly more powerful than *Cascading Style Sheets* [22], which can only specify how markup elements are rendered (font, color etc.) without allowing element reordering.

XSL works by recursively transforming XML documents into other formats such as HTML, according to a set of style rules augmented with a scripting language. XSL is based in part on a styling language for SGML, *Document Style Semantics and Specification Language* (DSSSL) [19]. The transformation process can take place inside an XML-enabled browser, or in an applet or script run by a browser without XML capabilities.

3. ARCHITECTURES FOR CLEARINGHOUSES

Two architectures are described. The first is a conventional implementation model for clearinghouses, which can be applied using established web development techniques. The second is a new architecture.

3.1 The Traditional Web Development Model

The main components in a traditional implementation of a catalog clearinghouse will generally include the users' web browsers (clients), the clearinghouse web server, a RDBMS, a fulfillment agent and the merchants' order acceptance points (telephones, fax machines, e-mail addresses, etc.). The user must first locate a particular catalog by entering keywords into a search form, which is posted to the web server. The server formats one or more SQL SELECT queries based on the search terms. The queries are sent to the RDBMS, where their execution will typically require relational joins on a number of tables in the database. Query results are returned to the web server.

Using a sequence of such queries, the web server determines the number of matching catalogs. If zero or multiple catalogs match, the user must continue searching. Once a single catalog is identified, the web server constructs an HTML order form dynamically. This action can be expensive. For instance, a restaurant menu involves many items in multiple categories (appetizers, entrées, desserts, beverages). The server must execute nested loops to construct an order form, usually embedded inside HTML tables. The order form is sent to the user's browser.

The user completes the order form and submits it to the web server. The server constructs additional SQL queries (which may now include SELECT, INSERT and UPDATE) that enter the order into the RDBMS. At this point, a fulfillment agent is responsible for transmitting the order to the appropriate merchant. If orders must be fulfilled in real-time (e.g., restaurants), the web server itself may act as the fulfillment agent, perhaps by relaying the order to a fax machine. In other cases (e.g., mail-order catalogs) a separate process can periodically extract new orders from the database for transmission to merchant order acceptance points.

A complete e-commerce implementation involves details not included in the above outline, including user session management (with HTTP cookies or hidden form fields), secure payment and transaction processing for fault-tolerance.

3.2 A New Architecture Based on XML and XSL

Representing catalog data using XML leads to an alternate implementation architecture, which offers the benefits outlined in the introduction. The new architecture extends the conventional one, and is outlined in Figure 1. Two databases are now used, an XML database for catalog information, and an RDBMS for order processing.

The workings of the new architecture are sketched in Figure 2. In step (1), the user submits a search form to the web server. Search interfaces for structured text documents can be quite powerful. For instance, the user can specify expressions with Boolean and comparison operators as well as hierarchical and containment restrictions, restricting expressions to only match within certain markup-element contexts. These contextual restrictions are much more precise than the word-distance based searches offered by typical web search engines. Again, although Boolean and comparison-based searches can be done with relational databases, they are likely to be less efficient than ones on text databases, which are indexed differently.

The completed search form is sent to the web server (2). The web server formulates a series of queries to the XML database (3), where a search occurs (4), eventually locating a single catalog (as before, the user may need to be consulted again if a unique catalog is not located). The catalog has XML and XSL components. These are returned to the web server (5), which does only *minimal* processing (6), e.g. to add user session information. The data is sent to the client browser (7), which transforms the XML into HTML by applying the XSL rule set (8). An example of this transformation is shown in section 3.3.

The browser now renders the catalog order form for the user. At this point, the process continues much as in the traditional model, using an RDBMS and a fulfillment agent (not shown). It is important to continue to use an RDBMS here because fault-tolerance is especially crucial once an order is placed. Transaction processing systems have clear model of failure events and a well-established track record of fault-tolerance.

In the new architecture, the fulfillment agent may query both the XML database and the SQL database for additional information (e.g. merchant address). This is because the order information entered into the RDBMS will only store catalog and item identifiers, to avoid duplicating data that is in the XML database (because duplicating information would violate general database design principles).

The benefits of the new implementation can now be reviewed. First, the web server's load is significantly reduced because formatting is done in the browser. Next, since each XML catalog may have a different XSL rule set, catalogs are rendered uniquely. In addition, powerful search techniques drawing on a long history of SGML research can be applied to perform searches that are easier to express and more efficient than is possible with relational data. For instance, XML-QL [14] allows for the specification of queries with conditions, structured text "joins", path regular expressions to handle hierarchy, nested queries, etc. Because XML-QL itself uses a syntax similar to that of XML documents, it is easier to express structured text queries in a language like XML-QL than in SQL.

Finally, it should be noted that the architecture can be modified to effect the XML to HTML transformation on the server, using transformation software such as XT [9]. While placing an additional load on the server, this solution can generate browser-independent HTML. This would allow developers to deploy solutions based on this architecture without waiting for XML/XSL-enabled browsers to become commonplace.

3.3 Sample XML and XSL Representations

In this section, sample XML and XSL data for a "restaurant clearinghouse" are shown. These are simplified for clarity and brevity, e.g., the restaurant DTD shown only stores one restaurant menu instead f many. The DTD for a restaurant includes its address and its menu. The menu has sections for appetizers, entrées, desserts and beverages.

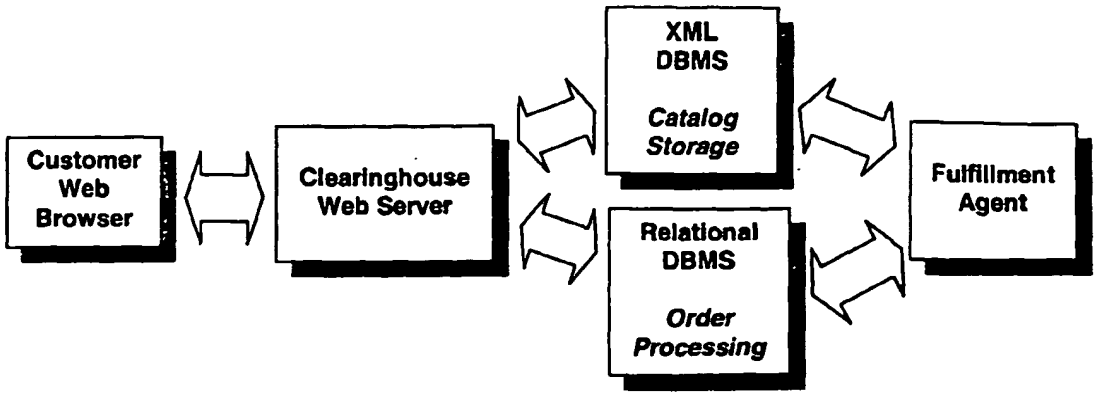

Figure 1: Clearinghouse Architecture Outline

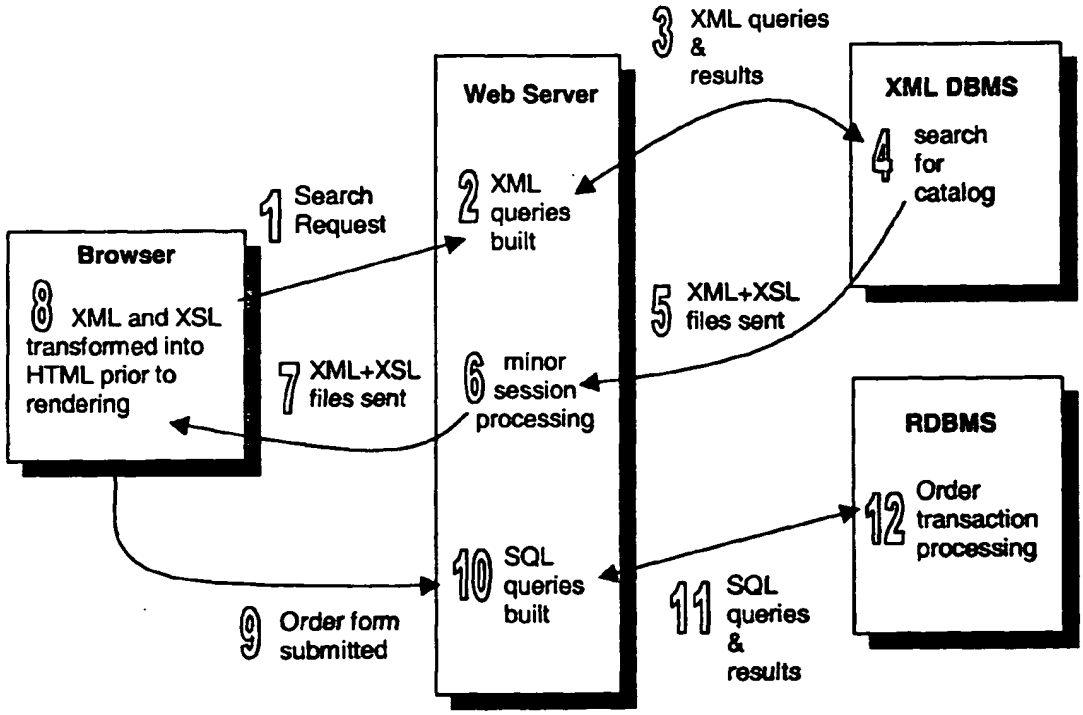

Figure 2: Clearinghouse Architecture Detail

Each section contains one or more menu items. A restaurant must offer at least one entrée, but may offer zero or more items in each of the other categories. A menu item consists of a description, a price and a "low fat" flag. An XML DTD for these conditions is shown in Figure 3. The EBNF notation uses a combination of common regular expression sequencing operators and context-free grammar productions. For instance, the operators "*", "+" and "?" denote repetitions of

"zero or more", "one or more" and "zero or one" times respectively; the "|" operator denotes "or" (alternation) and the "." operator denotes sequencing. "#PCDATA" denotes (essentially) XML text.

Figure 4 shows fragments of a data file conforming to the DTD. Note that all tags are properly nested and appear in pairs, except the "<lowfat/>" tag which has a content model of EMPTY in the DTD, and hence can appear alone with the trailing "/". The file is indented for clarity, but in practice is stored according to XML rules for processing whitespace.

Figure 5 shows a minimal XSL style sheet for transforming the above XML data into ordinary HTML. The style sheet specifies four transformation rules. Each transformation rule contains a pattern and an action. The XML document's natural parse tree structure is traversed to find nodes that match the pattern part of a rule. At matching nodes, the action part is used to derive a transformed sub-tree, which is attached at the current node (replacing the one already there). This process continues recursively until no patterns match. Therefore, transformations can be specified compactly without resorting to looping constructs. Furthermore, the W3C design goals state that XSL should leverage the power of scripting languages. This means that programmers more comfortable with a procedural paradigm will be able to use loops to effect the above transformation. Scripting languages that make tree data available to the programmer for use in calculations, decisions and loops can perform arbitrary restructuring.

The style sheet in Figure 5 is hand-coded according to the current W3C working draft for XSL [8], and tested using XT [9], a Java-based transformation engine for XSL. Earlier versions were tested in Microsoft's Technology Preview XSL processor [24], an ActiveX control. The style sheet transforms a restaurant XML document into a simple HTML file where all menu items (from all categories joined together) are listed as an HTML table. HTML tags embedded in the XSL rules are shown in uppercase for clarity. The style sheet in Figure 5 contains a total of four rules.

While XSL style sheets are not very amenable to hand coding, the abundance of graphical development tools indicates that style sheet generators will soon be routinely available (some already exist, e.g. [25]). Once a style sheet is created, a transformation engine such as XT [9] recursively applies the XSL rules to transform the sample XML document into HTML, as shown in Figure 6.

4. IMPLEMENTATION APPROACHES AND FUTURE WORK

The XML-based clearinghouse architecture is designed to be implemented efficiently using primarily off-the-shelf components. Available components range from inexpensive choices for prototyping, to more expensive choices for deployment. Many XML and XSL resources are listed in [11] (an excellent starting point for learning XML). Many software tools for structured documents are also listed at [20]; space limits mention here to a few.

Web server activities can take place in-process or out-of-process. Within the server process, server-side scripting and dynamically linked libraries (i.e., server *Application Program Interfaces*) provide choices between simplicity (scripting) and performance (APIs). Alternately, the server can initiate additional worker processes using the older Common Gateway Interface (CGI/bin) techniques. Server-side scripting is a good compromise between efficiency and complexity, and is supported by leading web servers, including Apache (*PHP/FI* scripting), Microsoft (*VBScript/JScript*) and Netscape (*Javascript*).

RDBMS choices range from simply placing the database on the web server's host, where the server accesses them directly using the appropriate libraries (e.g. *ODBC* compliant drivers), to multi-tiered solutions involving separate servers. A separate RDBMS server is preferable for many reasons. Choices range from the inexpensive *MySQL* to high-end enterprise servers from Oracle, IBM, Microsoft etc., which support transaction processing. A wide variety of e-commerce code is available to implement ordering and payment.

Tools for XML and XSL are rapidly appearing, based on extensive SGML development. Search engines like PAT [16] have shown that high performance is achievable on very large databases. A public domain choice is *SARA* [6]. These search engines have been extensively used with enormous textual databases and offer the advanced search features described earlier.

Intriguing techniques blending relational and structured-text theory are emerging, which should lead to additional choices for XML databases. In addition, new query languages have appeared, e.g. SGML-QL [17] and XML-QL [14]. Search engines for these should follow. Also, XML processors in Java and other languages [11] are suitable for developing prototype search engines (currently, Java run-time performance may be inadequate for high-volume production use at the server side).

XSL software suitable for prototyping and development (but not for deployment, yet) is available. These include XML Styler [25] for XML-based style sheet generation and the XT processor [9] for XSL transformations. Other Java-based XSL tools [11] and SGML styling tools [20] are available as well.

For this paper, work is under way on a prototype implementation using some of the tools mentioned above, and on additional design issues. The latter include enabling merchants to update their data and presentation online (probably using Java), developing flexible web search interfaces based on current SGML interfaces or Java *displets* [7], tracking customer information, etc.

Additional work is also needed on applying database validation rules and type checking to XML's textual data, and efficiently performing SQL-like aggregate queries on XML data for reports and data mining. These problems will probably involve the application of techniques integrating relational and structured data or relational variants [1,4].

Acknowledgements

The author thanks the anonymous referees for insightful comments on this paper.

5. REFERENCES

[1] Abiteboul, S. and Bidoit, N., Non first normal form relations: An algebra allowing data restructuring. *Journal of Computer and System Sciences*, 33(3), 361-393 (1986).

[2] Aho, A. V., Sethi, R. and Ullman, J.D., *Compilers: Principles, Techniques, and Tools*. Reading:, Addison-Wesley, 1986, rpt. corr. (1988).

[3] Berg, D.L., Gonnet, G. H. and Tompa, F. W., The New Oxford English Dictionary Project at the University of Waterloo, Technical Report OED-88-01. University of Waterloo Centre for the New Oxford English Dictionary (1988).

[4] Blake G.E., Consens M.P., Kilpeläinen, P., Larson, P.-A., Snider, T., and Tompa, F.W., Text/Relational Database Management Systems: Harmonizing SQL and SGML, *Proceedings of Applications of Databases (ADB-94)*, Vadstena, 267-280 (1994).

[5] Bray, T., Paoili, J., and Sperberg-McQueen, C. M., (ed) Extensible Markup Language (XML) 1.0, World Wide Web Consortium Recommendation 10-February-1998 (1988).

[6] Burnard, L., Introducing *SARA*: an SGML-Aware Retrieval Application for the British National Corpus, *Second Conference on Teaching and Language Corpora*, Lancaster University (1996).

[7] Ciancarini, P., Rizzi, A. and Vitali, F., An extensible rendering engine for XML and HTML, *Proceedings of the Seventh International WWW Conference*, Brisbane. Also in *Computer Networks and ISDN Systems*, 30, issues 1-7 (1998).

[8] Clark, J. and Deach, S. (ed), Extensible Stylesheet Language (XSL), Version 1.0, World Wide Web Consortium Working Draft 18-August-1998 (1998).

[9] Clark, J. XT (Java transformation engine for XSL). http://jclark.com/xml/xt.html (1998).

[10] Codd, E.F., Further Normalization of the Data Base Relational Model. *In Data Base Systems*, Courant Computer Science Symposia Series 6. Englewood Cliffs, NJ, Prentice-Hall (1972).

[11] Cover, R., Extensible Markup Language. A very large repository of XML and XSL information, at the *Summer Institute of Linguistics*. http://www.sil.org/sgml/xml.html (1997).

[12] Cyberchefs Home Page. http://www.cyberchefs.com/ (1998).

[13] Cybermeals Home Page. http://www.cybermeals.com/ (1998).

[14] Deutsch , A., Fernandez , M., Florescu, D., Levy, A., and Suciu, D., XML-QL: A Query language for XML, Submission to the World Wide Web Consortium 19-August-1998 (1998).

[15] Goldfarb, C. F., A Generalized Approach to Document Markup, *ACM SIGPLAN Notices* 16(6), 68-73 (1981).

[16] Gonnet, G.H., Baeza-Yates, R. A., Snider, T.: Lexicographical Indices for Text: Inverted files vs. PAT trees, Technical Report OED-91-01. University of Waterloo Centre for the New Oxford English Dictionary (1991).

[17] Harié , S., Le Maitre , J., Murisasco, E. and Véronis , J., The MULTEXT SgmlQL Query Language Reference, Multilingual Text Tools and Corpora Project, Centre National de la Recherche Scientifique, http://www.lpl.univ-aix.fr/projects/SgmlQL/ (1997).

[18] ISO 8879:1986, Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML). International Organization for Standardization. Ref. No. ISO 8879:1986 (E). Geneva/New York (1986).

[19] ISO/IEC DIS 10179.2:1994. Information Technology - Text and Office Systems - Document Style Semantics and Specification Language (DSSSL). International Organization for Standardization/ International Electrotechnical Commission. Geneva (1994).

[20] Kuikka, E. and Nikunen, E., Systems for Structured Documents (large list of software), http://www.cs.uku.fi/~kuikka/Systems97/systems97.html (1998).

[21] Lie, H. and Bos, B., Cascading Style Sheets, level 1, World Wide Web Consortium Recommendation 17-December-1996 (1996).

[22] Matzen, R.W. and Hedrick, G.E., A New Tool for SGML with Applications for the World Wide Web, *Proceedings of the 1998 ACM/SIGAPP Symposium on Applied Computing*, 752-759 (1998).

[23] Raymond, D. R., Flexible Text Display with Lector, *Computer*, August, 49-60 (1992).

[24] The *Microsoft XSL Processor*, Technology Preview Release, Microsoft Corporation, http://www.microsoft.com/xml/xsl/msxsl.asp (1998).

[25] *XML Styler*, ArborText Inc., http://www.arbortext.com/xmlstyler/index.htm (1998).

```
<?xml version="1.0"?>
<!DOCTYPE restaurant        [
<!ELEMENT restaurant        (name, address, menu)>
<!ELEMENT name              (#PCDATA)>
<!ELEMENT address           (street, city, (state | region), zip, phone)>
<!ELEMENT menu              (appetizers, entrees, desserts, beverages)>
<!ELEMENT street            (#PCDATA)>
<!ELEMENT city              (#PCDATA)>
<!ELEMENT state             (#PCDATA)>
<!ELEMENT region            (#PCDATA)>
<!ELEMENT zip               (#PCDATA)>
<!ELEMENT phone             (#PCDATA)>
<!ELEMENT appetizers        (item*)>
<!ELEMENT entrees           (item+)>
<!ELEMENT desserts          (item*)>
<!ELEMENT beverages         (item*)>
<!ELEMENT item              (description, price, lowfat?)>
<!ELEMENT description       (#PCDATA)>
<!ELEMENT price             (#PCDATA)>
<!ELEMENT lowfat            EMPTY>
]>
```

Figure 3: Sample XML DTD

```
<restaurant>
  <name>Toad Hall Cafe</name>
  <address>
    <street>116 Old Canton Road</street>
    <city>Boston</city>
    ( ... state, zip and phone ... )
  </address>
  <menu>
    <appetizers>
      <item>
        <description>cheese fritters</description>
        <price>4.99</price>
      </item>
      <item>
        <description>shrimp platter</description>
        <price>5.49</price>
        <lowfat/>
      </item>
    </appetizers>
    <entrees>
      <item>
        <description>grilled duck breast</description>
        <price>12.75</price>
        <lowfat/>
      </item>
    </entrees>
    (... desserts and beverages ...)
  </menu>
</restaurant>
```

Figure 4: Sample XML Document

```
<xsl:stylesheet  result-ns="html">

<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>
<xsl:process select="restaurant/name"/>
</TITLE>
</HEAD>
<BODY>
<H1>
Menu for
<xsl:process select="restaurant/name"/>
</H1>
<HR/>
<BR/>
<TABLE>
<xsl:process select="restaurant/menu"/>
</TABLE>
</BODY>
</HTML>
</xsl:template>

<xsl:template match="item">
<TR>
<xsl:process-children/>
</TR>
</xsl:template>

<xsl:template match="description">
<TD>
<xsl:process-children/>
</TD>
</xsl:template>

<xsl:template match="price">
<TD>
<B>
<xsl:process-children/>
</B>
</TD>
</xsl:template>

</xsl:stylesheet>
```

Figure 5: XSL Style Sheet

```
<HEAD>
<TITLE>
Toad Hall Café
</TITLE>
</HEAD>
<BODY>
<H1>
Menu for Toad Hall Cafe
</H1>
<HR/>
<BR/>
<TABLE>
<TR>
<TD>
cheese fritters
</TD>
<TD>
<B>
4.99
</B>
</TD>
</TR>
<TR>
<TD>
shrimp platter
</TD>
<TD>
<B>
5.49
</B>
</TD>
</TR>
<TR>
<TD>
grilled duck breast
</TD>
<TD>
<B>
12.75
</B>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

Figure 6: XML Transformed to HTML